Attorney Docket No.: 42390.P11422 *PATENT*

UNITED STATES PATENT APPLICATION

For

A METHOD FOR USING NON-TEMPORAL STREAMING STORES TO IMPROVE GARBAGE COLLECTION ALGORITHM

INVENTORS:

SREENIVAS SUBRAMONEY

RICHARD L. HUDSON

Prepared By:
BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
12400 WILSHIRE BOULEVARD
SEVENTH FLOOR
LOS ANGELES, CA 90025-1026

(408) 720-8300

"Express Mail" mailing label number: EL672751164US
Date of Deposit: June 19, 2001
I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" under 37 C.F.R. § 1.10 on the date indicated above and that this paper or fee has been addressed to the Assistant Commissioner for Patents, Washington, D. C. 20231
Maureen R. Pettibone
(Typed or printed name of person mailing paper or fee)
Maureen R. Pettibone
(Signature of person mailing paper or fee)
6/19/01
(Date signed)

A METHOD FOR USING NON-TEMPORAL STORES TO IMPROVE GARBAGE COLLECTION ALGORITHM

FIELD OF THE INVENTION

[0001] This invention relates generally to memory management in run-time environments, and more specifically to a garbage collection algorithm that uses non-temporal stores to reduce garbage collection time.

BACKGROUND OF THE INVENTION

[0002] The random access memory (RAM) of a computing system is a fixed size resource; currently a RAM size of 32 megabytes (Mb) is typical. The RAM must be managed properly to maintain system performance. In run-time environments such as Java or Microsoft CLI, memory management is handled by the system. Memory management includes a process known as "garbage collection". Garbage collection is a process with the aim of being as unobtrusive as possible in recycling memory. When a computer program is running it allocates and uses portions of memory on an ongoing basis. At some point the program may no longer need to use a particular portion of memory, e.g., the memory was allocated for a particular purpose that is no longer relevant. The portions that are no longer being used (garbage) are identified (collected) so that they can be reclaimed for future allocation. The garbage collection process taxes the central processing unit (CPU) and degrades system performance as perceived by the application. It is, therefore, highly desirable to reduce the time taken to reclaim unused portions of memory.

[0003] Typical computing systems have a cache memory between the CPU and main memory. The cache is small, typically 2 Mb or less, compared to main memory, that is typically 128 Mb. The cache is used to store, and provide fast access to data from

the most recently used memory locations. The data is brought to cache with the expectation that it may be accessed again soon. Garbage collection takes place in main memory, but because most programs operate under the assumption that recently accessed data may be accessed again soon, the processing of garbage collection takes place in the cache as described below.

[0004] A popular garbage collection algorithm for use in run-time environments is the moving garbage collection algorithm (MGCA). The MGCA examines a memory block that may typically be from 1 Mb to 4 gigabytes (Gb) in size. The MGCA determines which memory data from the block is in use (live data) and which is garbage. As the name implies, MGCAs move all live data to new consecutive memory locations. This compacts the live data into a smaller space than when it was co-located with the garbage. Once the live data is copied to new locations the entire block can be reclaimed and reallocated.

[0005] A typical MGCA has three phases: mark, repoint, and copy. In the mark phase the live objects, those to be moved to a new memory location, are determined. At this point new memory locations for the data objects are determined. In the repoint phase the live objects are examined and their references are changed so that they refer to new memory locations. In the copy phase, the contents of each live object are copied to the new memory location.

[0006] In many programs when data is accessed, for example to be copied, the data is brought into cache memory. As described above, the cache provides quick access to frequently used memory, and it is assumed that recently accessed data may need to be accessed again soon. If the data is not used again soon it is then deleted from the cache. This process, based on temporal access patterns, frequently results in data being stored to

cache only to be deleted when it is not accessed soon. This process taxes the cache memory in determining which data may be deleted from cache and also in having to actually delete it and possibly write back changed data to main memory.

[0007] When a live data object is copied to the new memory location, the data copied to the new memory location will not need to be accessed in the future. Therefore, copying the data to the cache in expectation of the data being accessed soon needlessly taxes CPU/cache resources.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] The present invention is illustrated by way of example and not intended to be limited by the figures of the accompanying drawings in which like references indicate similar elements and in which:

[0009] **Figure 1** is an illustration of an exemplary computing system for implementing the moving garbage collector of the present invention; and

[0010] **Figures 2A and 2B** describe the use of the non-temporal streaming stores feature of the CPU to reduce the time required for garbage collection.

DETAILED DESCRIPTION

[0011] An improved moving garbage collection algorithm is described. The algorithm allows efficient use of non-temporal stores to reduce the required time for garbage collection. Non-temporal stores (or copies) are a CPU feature that allows the copy of data objects within main memory with no interference or pollution of the cache memory. The live objects copied to new memory locations will not be accessed in the near future and therefore need not be copied through the cache. If implemented, this avoids copy operations and avoids taxing the hardware. The algorithm of the present invention uses the fact that live data objects will be stored to consecutive new memory locations in order to perform streaming copy procedures. Since each copy procedure has an associated CPU overhead, the process of streaming the copies reduces the degradation of system performance and thus reduces the overall time for garbage collection.

[0012] **Figure 1** is a diagram illustrating an exemplary computing system 100 for implementing the MGCA of the present invention. The use of non-temporal copy features and streaming copies for more efficient garbage collection described herein can be implemented and utilized within computing system 100. System 100 can represent a general-purpose computer, portable computer, or other like device. The components of computing system 100 are exemplary in which one or more components can be omitted or added. For example, one or more memory devices can be utilized for computing system 100.

[0013] Referring to **Figure 1**, computing system 100 includes a central processing unit 102 and a signal processor 103 coupled to a display circuit 105, main memory 104, static memory 106, and mass storage device 107 via bus 101. Computing system 100 can

also be coupled to a display 121, keypad input 122, cursor control 123, hard copy device 124, input/output (I/O) devices 125, and audio/speech device 126 via bus 101.

[0014] Bus 101 is a standard system bus for communicating information and signals. CPU 102 and signal processor 103 are processing units for computing system 100. CPU 102 or signal processor 103 or both can be used to process information and/or signals for computing system 100. CPU 102 includes a control unit 131, an arithmetic logic unit (ALU) 132, and several registers 133, which are used to process information and signals. Signal processor 103 can also include similar components as CPU 102.

[0015] Main memory 104 can be, e.g., a random access memory (RAM) or some other dynamic storage device, for storing information or instructions (program code), which are used by CPU 102 or signal processor 103. Main memory 104 may store temporary variables or other intermediate information during execution of instructions by CPU 102 or signal processor 103. Static memory 106, can be, e.g., a read only memory (ROM) and/or other static storage devices, for storing information or instructions, which can also be used by CPU 102 or signal processor 103. Mass storage device 107 can be, e.g., a hard or floppy disk drive or optical disk drive, for storing information or instructions for computing system 100.

[0016] Display 121 can be, e.g., a cathode ray tube (CRT) or liquid crystal display (LCD). Display device 121 displays information or graphics to a user. Computing system 100 can interface with display 121 via display circuit 105. Keypad input 122 is an alphanumeric input device with an analog to digital converter. Cursor control 123 can be, e.g., a mouse, a trackball, or cursor direction keys, for controlling movement of an object on display 121. Hard copy device 124 can be, e.g., a laser printer, for printing

information on paper, film, or some other like medium. A number of input/output devices 125 can be coupled to computing system 100.

[0017] The automated process of garbage collection in accordance with the present invention can be implemented by hardware and/or software contained within computing system 100. For example, CPU 102 or signal processor 103 can execute code or instructions stored in a machine-readable medium, e.g., main memory 104.

[0018] The machine-readable medium may include a mechanism that provides (i.e., stores and/or transmits) information in a form readable by a machine such as computer or digital processing device. For example, a machine-readable medium may include a read only memory (ROM), random access memory (RAM), magnetic disk storage media, optical storage media, flash memory devices. The code or instructions may be represented by carrier-wave signals, infrared signals, digital signals, and by other like signals.

[0019] As described above, a typical MGCA includes a copy phase in which live objects are copied to new memory locations. An embodiment of the present invention uses non-temporal streaming stores (NTSS) to complete the copy phase of garbage collection. The use of NTSS allows the completion of the copy phase in less time, resulting in faster garbage collection. Figure 2A describes the use of the non-temporal stores feature of the CPU to reduce the time required for garbage collection.

[0020] In **Figure 2A** data objects are shown as A through O. For example, data objects A, B, C, and D, are a root set of data (i.e., references within the CPU registers, for example). Then A, B, C, D, and all reachable (associated) data objects are considered live and will be moved. The arrows are references that indicate an association between the data objects. All transitive closure live objects are determined at the mark phase of

the MGCA. In one embodiment the mark phase may result in the creation of a mark stack as shown at 205. As shown in **Figure 2A**, data objects G and I are not included in the mark stack as they are not part of the root set nor are they reachable directly or transitively from members of the root set.

[0021] As each data object is determined to be live, new memory addresses are allocated. At the time a data object is added to the mark stack, a new memory location is determined for the data object because it is known that the data object must persist across the garbage collection. For example, as shown in **Figure 2B**, for each live data object referenced in the mark stack, a new memory location is allocated and designated. The mark stack is shown as 205B and a corresponding array of references to new memory locations is shown as 210B. At this point the references for each data object are changed. This is the repoint phase. For example, data object A references data objects J and K. The new reference for A will be A' and A' will reference J' and K'.

[0022] In the copy phase, the actual contents of data object A (the actual bits of data) will be copied to the new memory location referenced as A'. This includes the portion of data object A that references associated data. This portion may have already been updated, that is references to J and K have been changed to reference J' and K' respectively. In an alternative embodiment data object A is copied to the new memory location referenced as A' and then the data objects associated with A (i.e. J and K) are updated.

[0023] After data object A is copied to A', A' need not be accessed again. The component of garbage collection in regard to data object A is complete. The MGCA simply moves on to the next data object referenced in the mark stack and continues updating and moving. In the copy phase of the MGCA, the data object being copied (e.g.,

data object A) is brought into the cache. There is no need to copy the contents of A' to cache, however, because that data object (i.e., A') will not be accessed in the near future. The algorithm of the present invention, included as Appendix A, uses non-temporal stores (NTSs) to copy the data objects directly to the designated new location in memory. The CPU, in connection with the memory controller, executes a non-temporal store causing a copy of A to the A' memory location without interference or pollution of the cache memory. The use of NTS reduces the time necessary for garbage collection because it reduces the amount of data that is needlessly copied to the cache only to be deleted after some time.

[0024] The algorithm of the present invention also uses write combining (streaming copies) to reduce the time required for garbage collection. As shown in **Figure 2B**, the data objects referenced in mark stack 205B will be copied to new memory locations as referenced in mark stack 210B. For example, data objects A, B, and C will be copied to new locations as referenced by A', B', and C'. The algorithm of the present invention is designed such that all live objects are copied to consecutive new memory locations i.e., A', B', and C', for example, are consecutive locations. This allows use of the CPU's write combine (streaming copy) feature to copy several data objects asynchronously and transparently. Since each copy procedure has an associated CPU overhead, the process of streaming the copies reduces the degradation of system performance and thus reduces the time for garbage collection. The amount that can be copied is, of course, platform specific and dependent upon prevailing memory bus conditions.

[0025] In the foregoing specification, the invention has been described with reference to specific exemplary embodiments thereof. It will, however, be evident that

various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention as set forth in the appended claims. The specification and drawings are, accordingly, to be regarded in an illustrative sense rather than a restrictive sense.